
# Network Vulnerability Assessment Report

# Finding Severity Ratings

The following table defines levels of severity and corresponding CVSS score range that are used

throughout the document to assess vulnerability and risk impact.

| Severity | CVSS V3 Score Range | Definition |
|---|---|---|
| Critical | 9.0-10.0 | Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately. |
| High | 7.0-8.9 | Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible. |
| Moderate | 4.0-6.9 | Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved. |
| Low | 0.1-3.9 | Vulnerabilities are non-exploitable but would reduce an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window. |
| Informational | N/A | No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation. |

## Risk Factors

Risk is measured by two factors: Likelihood and Impact:

## Likelihood

Likelihood measures the potential of a vulnerability being exploited. Ratings are given based on the difficulty of the attack, the available tools, attacker skill level, and client environment.

### Impact

Impact measures the potential vulnerability's effect on operations, including confidentiality, integrity, and availability of client systems and/or data, reputational harm, and financial loss.

# Scope

| Assessment | Details |
| --- | --- |
| External Network Penetration Test | http://testfire.net/* |

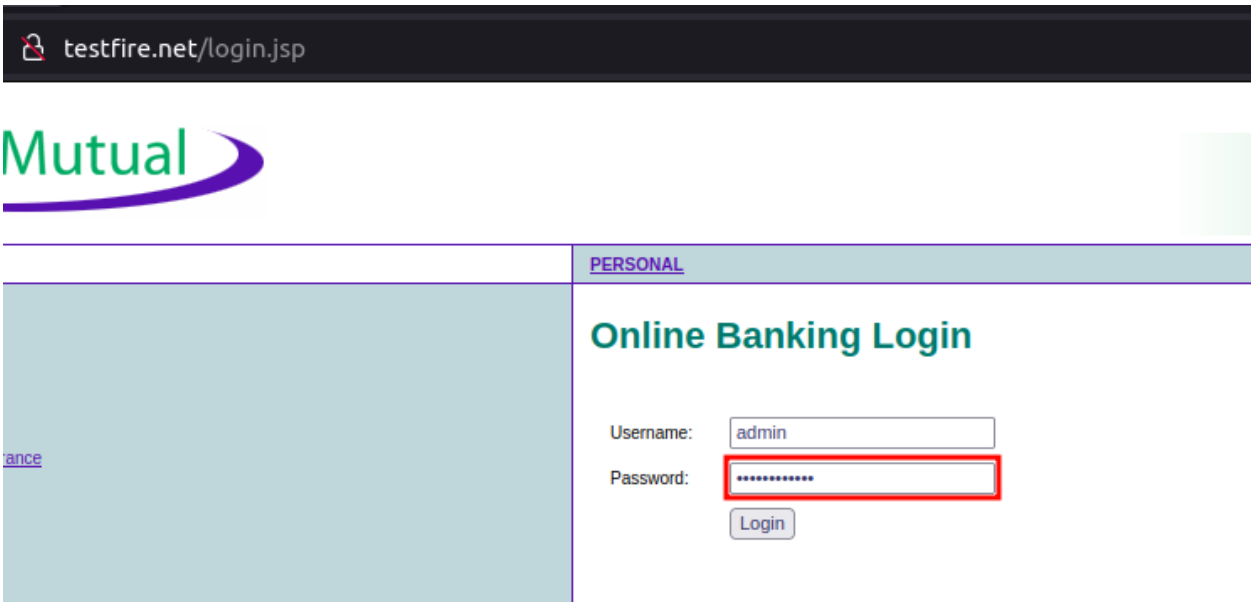
# Technical Findings

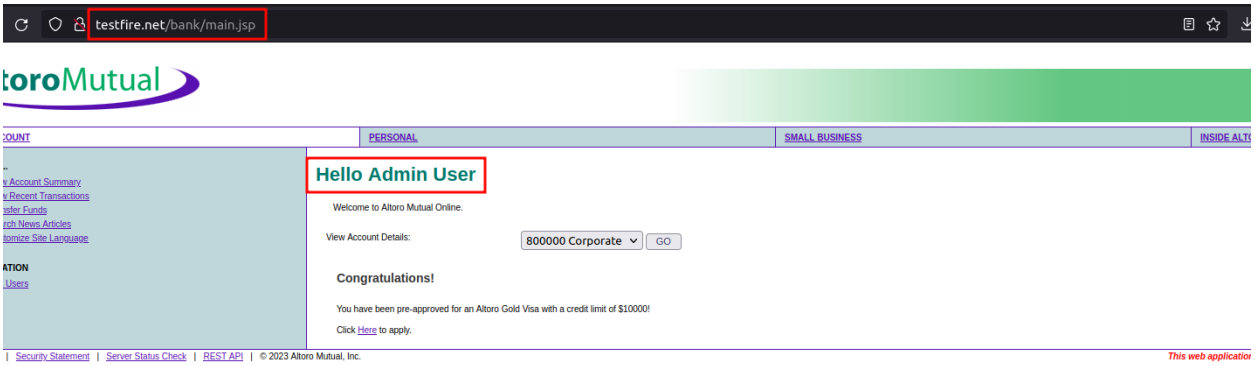
## Finding-001: Authentication Bypass using SQLi (Critical)

| | |
|---|---|
| Description: | Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query, leading to malicious code execution. Examples include SQL injection and OS command injection. |
| CEW Code: | CWE 89: SQL Injection |
| OWASP Category: | **A03:2021-Injection** slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition. |
| Affected URL: | http://testfire.net/login.jsp |
| Affected Parameter: | • uid (POST)<br>• passw (POST) |
| Payload Used: | ' or 1=1 --+ |
| References: | • https://owasp.org/www-community/attacks/SQL_Injection<br>• https://en.wikipedia.org/wiki/SQL_injection |

**Steps to reproduce:**

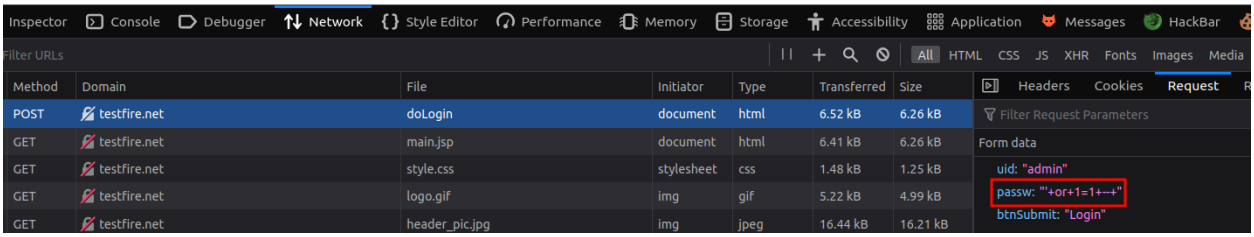
- Visit the affected URL and put the payload in the password field or both the username and password fields as shown below




website is published by IBM Corporation for the sole purpose of demonstrating the effectiveness of IBM products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is provided "as is" wit
ur use of this website. For more information, please go to http://www-142.ibm.com/software/products/us/en/subcategory/SW110.

2008, 2023, IBM Corporation, All rights reserved.



**Business Impact -** Extremely High

- Unauthorized Access: Attackers can gain access to sensitive data, user accounts, or administrative functionalities without valid credentials, compromising the confidentiality of data and systems.
- Data Breach: Sensitive information, such as customer records, financial data, or proprietary business data, could be exposed, leading to reputation damage, legal consequences, and financial losses.
- Compromised User Accounts: Users' accounts may be hijacked, leading to potential data manipulation, financial fraud, or unauthorized actions on behalf of the legitimate user.
- Loss of Trust: If customers' data is compromised due to authentication bypass, it can lead to a loss of trust and credibility in the affected business, resulting in decreased customer loyalty and potential loss of business.
- Regulatory Non-Compliance: Depending on the industry, a data breach resulting from authentication bypass might lead to violations of data protection regulations and laws, resulting in penalties and fines.
- Business Disruption: Detecting and mitigating the impact of an authentication bypass attack can lead to downtime and operational disruptions, affecting business continuity.
- Financial Costs: The cost of incident response, forensic investigations, system remediation, and potential legal actions can be substantial.
- Damage to Reputation: A successful attack can damage a company's reputation, leading to customer churn and difficulty in acquiring new customers.

- Competitive Disadvantage: If a business becomes known for security vulnerabilities, it may lose its competitive edge to competitors with better security practices.
- Additional Security Investments: After an incident, the organization might need to invest in enhanced security measures to prevent future attacks, which can incur additional expenses.

Overall, the impact of an "Authentication Bypass using SQLi" attack goes beyond just technical aspects, and it can have far-reaching consequences on a business's financial stability, reputation, and customer trust.

## Recommendations

- Prepared Statements: Use SQL prepared statements available in all web development languages and frameworks to avoid attacker being able to modify SQL query
- Character encoding: If you are taking input that requires you to accept special characters, encode it. Example. Convert all ‘ to \’ , “ to \”, \ to \\. It is also suggested to follow a standard encoding for all special characters such has HTML encoding, URL encoding etc

## Finding-002: Authentication Bypass using Brute Force attack (Critical)

| | |
|---|---|
| Description: | Identification and Authentication Failures refer to vulnerabilities in the authentication process, which can lead to unauthorized access and compromise of user accounts or sensitive information. This category encompasses various issues related to weak passwords, credential stuffing attacks, account enumeration, and other authentication-related weaknesses. |
| CEW Code: | CWE-799: Improper Control of Interaction Frequency |
| OWASP Category: | **A07:2021-Identification and Authentication Failures** category is still an integral part of the Top 10, but the increased availability of standardized frameworks seems to be helping. |
| Affected URL: | http://testfire.net/login.jsp |
| Affected Parameter: | • uid (POST)<br>• passw (POST) |
| Payload Used: | Seclists wordlists |
| References: | • https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/ |

**Steps to reproduce:**

- Run the below command, install any tool if necessary

```
ffuf -u 'http://testfire.net/doLogin' -H 'Content-Type: application/x-www-form-urlencoded' -d 'uid=USER&passw=PASS&btnSubmit=Login' -w passwords.txt:PASS -w usernames.txt:USER  -fs 0
```

## Business Impact - Extremely High

- Unauthorized Access: Brute force attacks can lead to unauthorized access to sensitive information, customer data, financial records, or intellectual

property, compromising data confidentiality.

- Data Breach: Successful authentication bypass can result in data breaches, damaging the company's reputation and leading to potential legal and financial liabilities.
- Service Disruption: Brute force attacks may overload authentication systems, causing service disruptions, downtime, and loss of productivity, impacting customer satisfaction and revenue generation.
- Account Takeover: If user accounts are compromised, attackers can gain control over legitimate users' accounts, leading to misuse, data manipulation, or financial fraud.
- Regulatory Compliance Issues: Data breaches resulting from authentication bypass can lead to non-compliance with data protection regulations, subjecting the organization to fines and penalties.
- Reputational Damage: News of a successful attack can tarnish the company's reputation, eroding customer trust and loyalty, resulting in loss of business and potential partners.
- Loss of Intellectual Property: For businesses reliant on proprietary technology or processes, an authentication bypass could lead to theft of valuable intellectual property.
- Operational Costs: Recovering from a successful attack involves investigation, incident response, and system remediation, incurring additional operational costs.
- Legal Consequences: In some cases, compromised data may lead to legal actions from affected parties, further escalating financial and legal

repercussions.

- Competitive Advantage Loss: A security breach can cause customers to seek more secure alternatives, giving competitors a chance to gain an advantage.

To mitigate the impact, businesses should implement strong authentication mechanisms, enforce account lockout policies, and monitor for suspicious login attempts to detect and prevent brute force attacks. Regular security assessments and employee training can also help bolster the organization's security posture.

## Recommendations

- Account Lockout Policy: Implement an account lockout policy that temporarily locks out an account after a certain number of failed login attempts. This prevents attackers from repeatedly attempting to guess passwords.
- Strong Password Policy: Enforce a strong password policy that mandates the use of complex passwords, including a mix of uppercase and lowercase letters, numbers, and special characters. Discourage the use of common or easily guessable passwords.
- Multi-Factor Authentication (MFA): Implement MFA to add an extra layer of security. This could include factors such as one-time passwords sent to a user's phone, biometric verification, or hardware tokens.
- Rate Limiting: Implement rate limiting on login requests to restrict the number of login attempts allowed from a single IP address within a specified time frame.

- Captcha and Human Verification: Use captchas or other human verification mechanisms on login pages to differentiate between genuine users and automated bots attempting brute force attacks.
- Monitoring and Alerting: Implement real-time monitoring and alerting for suspicious login activities, such as multiple failed login attempts from the same IP address or unusual login patterns.
- Regular Security Audits: Conduct regular security audits and vulnerability assessments to identify and address potential weaknesses in the authentication process.
- Two-Factor Recovery: Implement a secure account recovery process that involves two-factor authentication to prevent unauthorized access during the recovery phase.
- Lockout Recovery Mechanism: Design a secure lockout recovery mechanism, so legitimate users can regain access to their accounts through a secure and verified process.
- Account Enumeration Prevention: Avoid exposing account existence information during the authentication process. Return generic error messages instead of specific messages that indicate whether an account exists.
- Continuous User Education: Educate users about the risks of weak passwords and the importance of adopting good password practices to enhance overall security awareness.

By following these recommendations, businesses can significantly reduce the risk

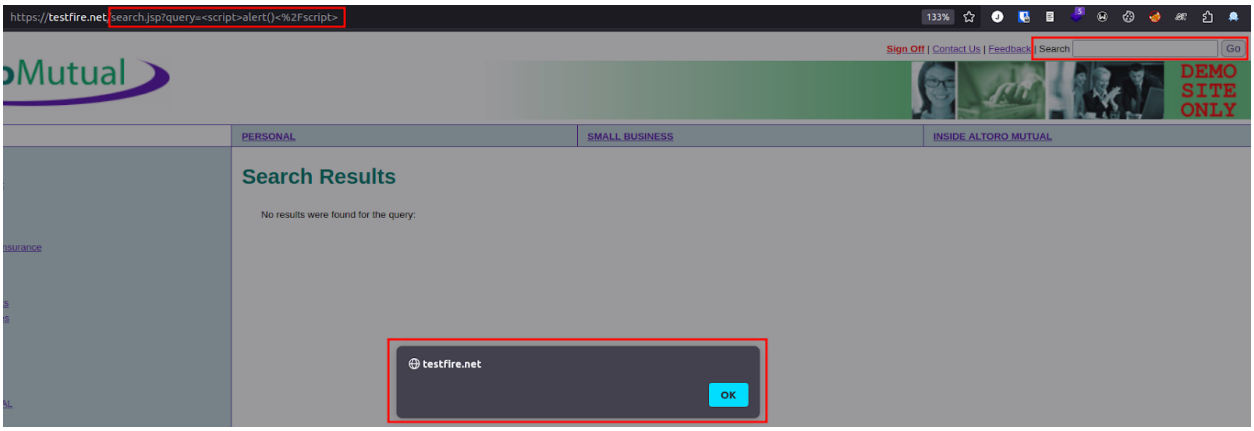
of successful authentication bypass through brute force attacks and enhance the security of their authentication mechanisms.

## Finding-003: Reflected XSS (Medium)

| | |
|---|---|
| Description: | Injection vulnerabilities refer to security flaws that occur when untrusted data is provided to an application and executed as code. Attackers exploit these vulnerabilities by injecting malicious code into the application, leading to unintended behavior, data manipulation, or unauthorized access. |
| CEW Code: | CWE 80: Cross-Site Scripting (XSS) |
| OWASP Category: | **A03:2021-Injection** slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition. |
| Affected URL: | • http://testfire.net/search.jsp |
| Affected Parameter: | • query (GET) |
| Payload Used: | <script>alert()</script> |
| References: | • https://owasp.org/Top10/A03_2021-Injection/<br>• https://owasp.org/www-community/attacks/xss/ |

### Steps to reproduce:

- Enter the payload in the text box and click the go button.



**Business Impact -** High

- Data Breach and Unauthorized Access: Reflected XSS allows attackers to inject malicious scripts into web pages viewed by other users, potentially leading to data breaches, unauthorized access to sensitive information, and exposure of confidential data.
- Compromised User Accounts: Attackers can steal session cookies or credentials through reflected XSS, leading to the compromise of user accounts, including those of customers, employees, or administrators.
- Loss of Customer Trust: Successful attacks can result in the exposure of sensitive customer data, leading to a loss of trust in the organization's security practices. This loss of confidence can lead to decreased customer loyalty, negative reputation, and potential customer churn.
- Financial Loss: If attackers gain access to financial data, such as credit card details, it can lead to fraudulent transactions and financial losses for both the business and its customers.
- Legal and Regulatory Consequences: Data breaches resulting from reflected XSS can lead to legal and regulatory consequences, such as fines or lawsuits for non-compliance with data protection regulations.
- Website Defacement and Downtime: Reflected XSS attacks may be used to deface the website, display inappropriate content, or disrupt the site's normal functionality, leading to downtime and loss of revenue for the business.
- Negative Publicity: A successful XSS attack can attract media attention and negative publicity, damaging the company's reputation and brand image.
- Resource Allocation: Dealing with the aftermath of an XSS attack requires

significant resources, including time, personnel, and financial investments, diverting focus from other essential business operations.

## Recommendations

- Input validation and sanitization: Ensure that all user-supplied input, such as data from query parameters, form fields, and URL fragments, is validated and sanitized before being used in the application. Sanitization should involve removing or escaping any potentially harmful characters.
- Output encoding: Encode all user-generated or dynamic content properly before displaying it back to users. Use the appropriate encoding methods depending on the context in which the data is being displayed (e.g., HTML entities encoding, JavaScript escaping, etc.).
- Use HTTP-only cookies: Use HTTP-only cookies to store sensitive data instead of storing them in JavaScript-accessible cookies. This prevents attackers from accessing cookies through client-side scripts, reducing the impact of XSS attacks.
- Implement Content Security Policy (CSP): CSP is a security feature that allows you to specify the domains from which various types of content can be loaded, including scripts. It can help mitigate the risk of XSS by restricting the sources of executable scripts.
- Use security libraries and frameworks: Leverage security libraries and frameworks that have built-in protections against XSS attacks. For example, frameworks like Angular, React, and Vue.js have mechanisms to handle output encoding and XSS prevention.
- Set the 'HttpOnly' and 'Secure' flags on cookies: By setting the 'HttpOnly' flag,

you prevent client-side scripts from accessing cookies. The 'Secure' flag ensures that cookies are only transmitted over encrypted (HTTPS) connections.

- Implement context-specific output encoding: Different contexts (e.g., HTML, JavaScript, CSS) have different rules for escaping special characters. Make sure to encode output data based on the specific context in which it will be rendered.
- Regular security testing: Conduct regular security assessments, including vulnerability scanning and penetration testing, to identify and fix potential XSS vulnerabilities proactively.
- Stay updated with security best practices: Keep yourself informed about the latest security best practices and trends to ensure you are using the most effective measures against XSS attacks.
- Educate developers: Train and educate developers on secure coding practices, particularly related to input validation, output encoding, and handling user-generated content.

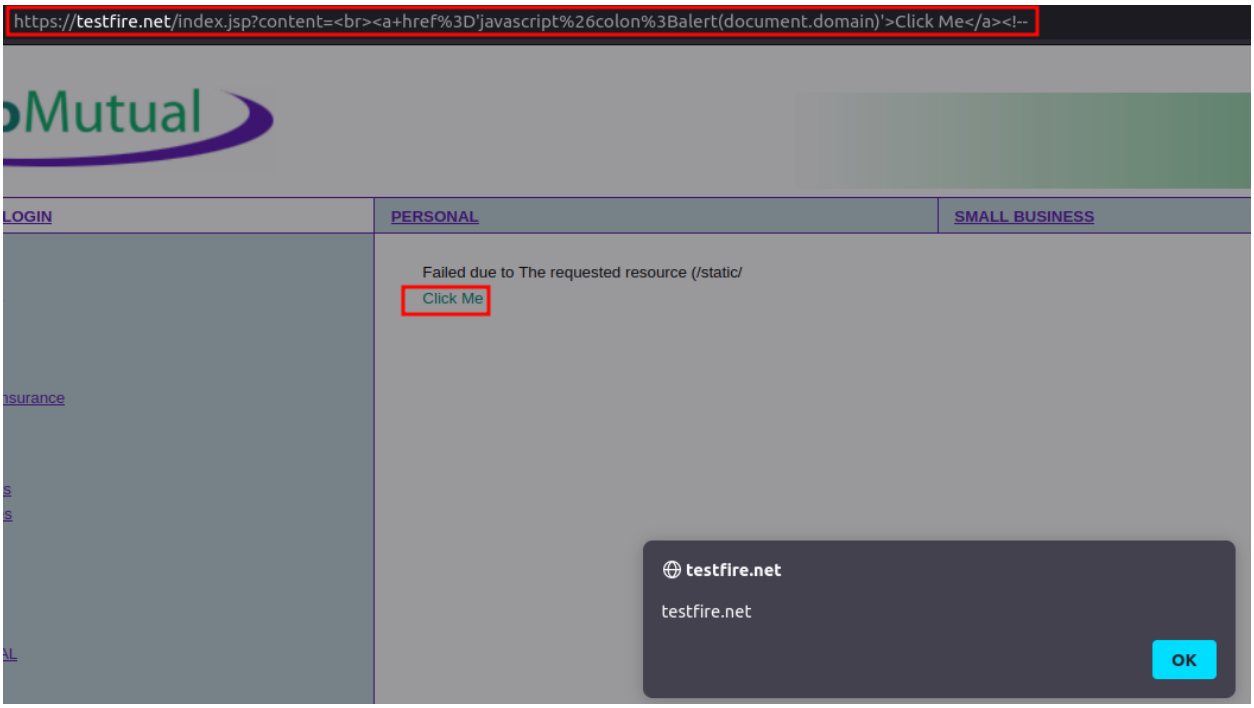
## Finding-003: Reflected XSS via HTML Injection (Medium)

| | |
|---|---|
| Description: | Injection vulnerabilities refer to security flaws that occur when untrusted data is provided to an application and executed as code. Attackers exploit these vulnerabilities by injecting malicious code into the application, leading to unintended behavior, data manipulation, or unauthorized access. |
| CEW Code: | CWE 80: Cross-Site Scripting (XSS) |
| OWASP Category: | **A03:2021-Injection** slides down to the third position. 94% of the applications were tested for some form of injection, and the 33 CWEs mapped into this category have the second most occurrences in applications. Cross-site Scripting is now part of this category in this edition. |
| Affected URLs: | • https://testfire.net/index.jsp |
| Affected Parameters: | • content (GET) |
| Payloads Used: | • <br><a+href%3D'javascript%26colon%3Balert(document.domain)'>Click Me</a><!-- |
| References: | • https://owasp.org/Top10/A03_2021-Injection/<br>• https://owasp.org/www-community/attacks/xss/ |

**Steps to reproduce:**

- Visit this URL:

  https://testfire.net/index.jsp?content=%3Cbr%3E%3Ca+href%3D%27javascript%26colon%3Balert(document.domain)%27%3EClick%20Me%3C/a%3E%3C!-- and click on “Click Me” link




**Business Impact -** High

- Data Breach and Unauthorized Access: Reflected XSS allows attackers to inject malicious scripts into web pages viewed by other users, potentially leading to data breaches, unauthorized access to sensitive information, and exposure of confidential data.
- Compromised User Accounts: Attackers can steal session cookies or credentials through reflected XSS, leading to the compromise of user accounts, including those of customers, employees, or administrators.
- Loss of Customer Trust: Successful attacks can result in the exposure of sensitive customer data, leading to a loss of trust in the organization's security practices. This loss of confidence can lead to decreased customer loyalty, negative reputation, and potential customer churn.
- Financial Loss: If attackers gain access to financial data, such as credit card details, it can lead to fraudulent transactions and financial losses for both the

business and its customers.

- Legal and Regulatory Consequences: Data breaches resulting from reflected XSS can lead to legal and regulatory consequences, such as fines or lawsuits for non-compliance with data protection regulations.
- Website Defacement and Downtime: Reflected XSS attacks may be used to deface the website, display inappropriate content, or disrupt the site's normal functionality, leading to downtime and loss of revenue for the business.
- Negative Publicity: A successful XSS attack can attract media attention and negative publicity, damaging the company's reputation and brand image.
- Resource Allocation: Dealing with the aftermath of an XSS attack requires significant resources, including time, personnel, and financial investments, diverting focus from other essential business operations.

## Recommendations

- Input validation and sanitization: Ensure that all user-supplied input, such as data from query parameters, form fields, and URL fragments, is validated and sanitized before being used in the application. Sanitization should involve removing or escaping any potentially harmful characters.
- Output encoding: Encode all user-generated or dynamic content properly before displaying it back to users. Use the appropriate encoding methods depending on the context in which the data is being displayed (e.g., HTML entities encoding, JavaScript escaping, etc.).
- Use HTTP-only cookies: Use HTTP-only cookies to store sensitive data instead of storing them in JavaScript-accessible cookies. This prevents attackers from accessing cookies through client-side scripts, reducing the

impact of XSS attacks.

- Implement Content Security Policy (CSP): CSP is a security feature that allows you to specify the domains from which various types of content can be loaded, including scripts. It can help mitigate the risk of XSS by restricting the sources of executable scripts.
- Use security libraries and frameworks: Leverage security libraries and frameworks that have built-in protections against XSS attacks. For example, frameworks like Angular, React, and Vue.js have mechanisms to handle output encoding and XSS prevention.
- Set the 'HttpOnly' and 'Secure' flags on cookies: By setting the 'HttpOnly' flag, you prevent client-side scripts from accessing cookies. The 'Secure' flag ensures that cookies are only transmitted over encrypted (HTTPS) connections.
- Implement context-specific output encoding: Different contexts (e.g., HTML, JavaScript, CSS) have different rules for escaping special characters. Make sure to encode output data based on the specific context in which it will be rendered.
- Regular security testing: Conduct regular security assessments, including vulnerability scanning and penetration testing, to identify and fix potential XSS vulnerabilities proactively.
- Stay updated with security best practices: Keep yourself informed about the latest security best practices and trends to ensure you are using the most effective measures against XSS attacks.
- Educate developers: Train and educate developers on secure coding

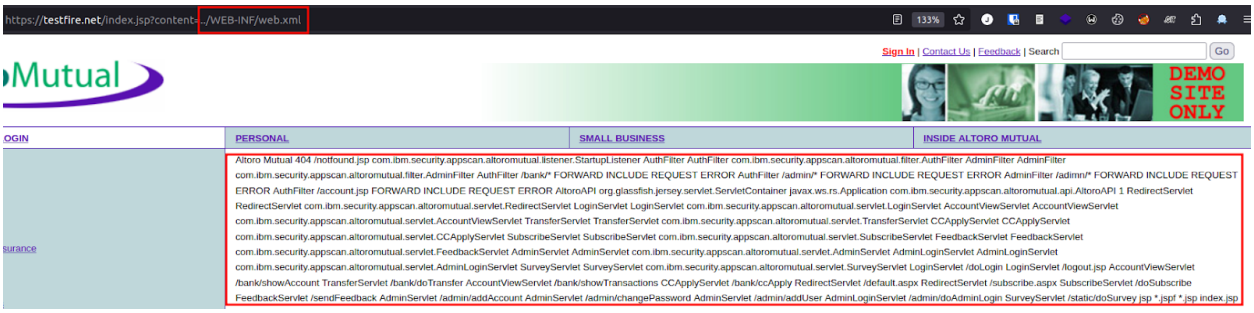
practices, particularly related to input validation, output encoding, and handling user-generated content.

## Finding-004: Local File Inclusion/Path Traversal (High)

| | |
|---|---|
| Description: | Security Misconfiguration refers to the improper implementation or configuration of security controls, which can leave an application or system vulnerable to attacks. This category includes a wide range of misconfigurations, such as default settings, unnecessary services or features enabled, and insufficient security settings. |
| CEW Code: | CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') |
| OWASP Category: | **A05:2021-Security Misconfiguration** moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category. |
| Affected URL: | https://testfire.net/index.jsp |
| Affected Parameter: | content (GET) |
| Payload Used: | ../WEB-INF/web.xml |
| References: | • https://owasp.org/www-community/attacks/Path_Traversal |

### Steps to reproduce:

- Visit this URL: https://testfire.net/index.jsp?content=../WEB-INF/web.xml




## Business Impact - High

- Data Exposure: Path traversal attacks can lead to unauthorized access to sensitive files and data, potentially exposing confidential customer information, financial records, or intellectual property.
- Reputation Damage: A successful attack can damage the organization's reputation, eroding customer trust and loyalty. News of a security breach can discourage new customers and partners from doing business with the company.
- Legal and Compliance Issues: If the organization handles sensitive data (e.g., personal or financial information), a breach due to path traversal may result in legal liabilities, fines, or penalties for non-compliance with data protection regulations.
- Downtime and Disruption: In some cases, the attack can cause service disruptions or website downtime, leading to lost revenue and productivity.
- Business Continuity: If critical files or resources are compromised, it can affect the business's ability to operate smoothly, impacting business continuity.
- Financial Loss: The costs associated with investigating and mitigating the attack, as well as potential legal fees, can result in financial losses for the organization.
- Competitive Advantage Loss: A security incident can damage the company's competitive advantage and open opportunities for competitors to gain market share.
- Customer Loss: Customers may lose trust in the organization's security measures, leading to churn and a decline in customer base.

## Recommendations

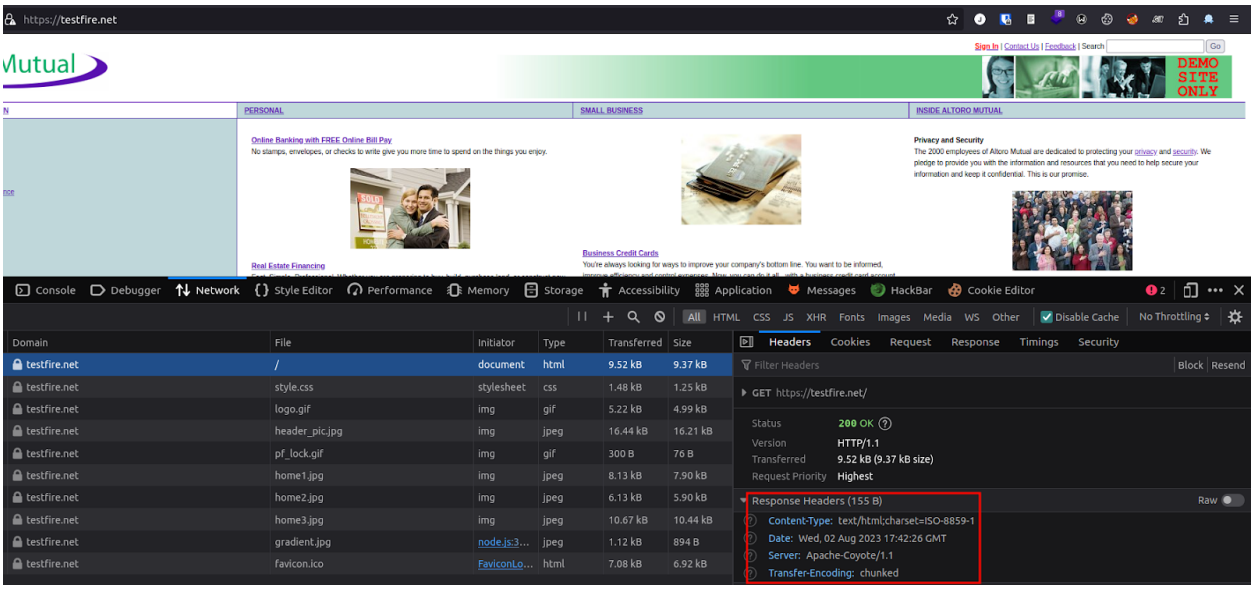
- Input Validation and Sanitization: Implement robust input validation and sanitization mechanisms in your web application. Validate and sanitize all user-supplied input, including file paths and parameters, to prevent attackers from injecting malicious characters or sequences that could lead to path traversal.
- File Access Controls: Enforce strict access controls on files and directories within your application. Limit access to only those files and directories that are explicitly required for normal application functionality. Avoid using user-supplied input directly as file paths, and instead, use a whitelist approach to allow only specific, trusted file paths to be accessed.
- Use Framework Security Features: If you are using a web framework or a library, make sure to leverage its built-in security features to prevent path traversal attacks. Many modern frameworks offer security functionalities like secure file handling and path resolution that can mitigate the risk of such attacks.

## Finding-005: Missing HTTP Security Headers (Low)

| | |
|---|---|
| Description: | Security Misconfiguration refers to the improper implementation or configuration of security controls, which can leave an application or system vulnerable to attacks. This category includes a wide range of misconfigurations, such as default settings, unnecessary services or features enabled, and insufficient security settings. |
| CEW Code: | CWE-693: Protection Mechanism Failure |
| OWASP Category: | **A05:2021-Security Misconfiguration** 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category. |
| Affected URL: | http://testfire.net |
| References: | • https://kcm.trellix.com/corporate/index?page=content&id=KB90241<br>• https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Headers_Cheat_Sheet.html |

**Steps to reproduce:**

- Use browser's  network tab to view HTTP response headers




- Nuclei tool output:

```
[http-missing-security-headers:referrer-policy] [http] [info] http://testfire.net
[http-missing-security-headers:clear-site-data] [http] [info] http://testfire.net
[http-missing-security-headers:cross-origin-embedder-policy] [http] [info] http://testfire.net
[http-missing-security-headers:cross-origin-opener-policy] [http] [info] http://testfire.net
[http-missing-security-headers:content-security-policy] [http] [info] http://testfire.net
[http-missing-security-headers:x-frame-options] [http] [info] http://testfire.net
[http-missing-security-headers:x-content-type-options] [http] [info] http://testfire.net
[http-missing-security-headers:cross-origin-resource-policy] [http] [info] http://testfire.net
[http-missing-security-headers:strict-transport-security] [http] [info] http://testfire.net
[http-missing-security-headers:permissions-policy] [http] [info] http://testfire.net
[http-missing-security-headers:x-permitted-cross-domain-policies] [http] [info] http://testfire.net
```

## **Business Impact -** Low

- Increased Security Risks: Without proper security headers, the application is more vulnerable to attacks like Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Clickjacking, and others. These attacks can lead to the compromise of sensitive user data, unauthorized access, and potential

legal liabilities.

- Data Breach: Missing security headers can make it easier for attackers to exploit vulnerabilities in the application, leading to data breaches. A data breach can result in significant financial losses, loss of customer trust, and damage to the brand's reputation.
- Regulatory Non-Compliance: Many data protection regulations and industry standards require the implementation of certain security measures, including specific HTTP security headers. Failure to comply with these regulations can result in legal consequences, fines, and other penalties.
- Loss of Customer Trust: Customers value their privacy and security when using web applications. If they perceive an application as insecure or not taking appropriate security measures, they may lose trust in the organization, leading to decreased user engagement and potential loss of customers.
- Negative Publicity and PR Crisis: Security breaches and vulnerabilities in an application can attract negative media attention and create a PR crisis for the business. Managing such incidents can be time-consuming, costly, and may harm the company's image in the long term.
- Disruption of Services: Successful attacks exploiting the absence of security headers can lead to the disruption of services, affecting the availability and functionality of the application. Downtime can result in lost revenue and dissatisfied customers.
- Competitive Disadvantage: Security-conscious customers may prefer competitors that prioritize and demonstrate a commitment to robust security practices. The absence of security headers can put the organization at a

competitive disadvantage.

- Increased Support and Recovery Costs: Dealing with security incidents, identifying vulnerabilities, and mitigating the impact can be expensive in terms of resources, time, and effort. Properly implementing security headers can reduce the risk of such incidents and their associated costs.

## Recommendations

- Enable Content Security Policy (CSP): Implement a robust CSP that restricts which content sources are allowed to be loaded and executed, mitigating the risk of code injection attacks like XSS. Regularly review and fine-tune the policy to ensure it does not interfere with legitimate functionality.
- Use Strict-Transport-Security (HSTS): Enable HSTS to enforce secure connections over HTTPS only, reducing the risk of man-in-the-middle attacks. Set a reasonable and appropriate max-age directive to maintain HSTS for a specified duration.
- Deploy X-XSS-Protection Header: Enable the X-XSS-Protection header to enable the XSS filter in the browser, providing an additional layer of protection against certain types of XSS attacks.
- Implement X-Content-Type-Options: Enable the X-Content-Type-Options header with the value "nosniff" to prevent MIME sniffing and enforce the declared content type.
- Use X-Frame-Options: Implement the X-Frame-Options header with the value "deny" or "sameorigin" to prevent clickjacking attacks by controlling how the page can be embedded in frames or iframes.

- Leverage Content-Security-Policy-Report-Only: Initially, deploy the Content-Security-Policy-Report-Only header to observe its impact without blocking any resources. Monitor the reports and fine-tune the policy accordingly before moving to full enforcement.
- Consider Referrer-Policy: Set the Referrer-Policy header to control what information about the current page is included in the Referer header when navigating to other pages.
- Perform Regular Security Scans and Penetration Testing: Conduct regular security scans and penetration testing to identify vulnerabilities, including any missing HTTP security headers. Address the identified issues promptly.
- Stay Informed About Security Best Practices: Stay updated with the latest security best practices and recommendations. Follow the OWASP guidelines and other reputable security resources to enhance web application security.
- Implement Secure Development Practices: Encourage secure coding practices within the development team. Conduct security training and workshops to raise awareness about security risks and measures.
- Maintain Software and Libraries: Keep all software, frameworks, and libraries up to date with the latest security patches and updates.
- Use Security Headers in All Web Pages: Ensure that security headers are consistently implemented across all pages of the application, not just specific sections.
- Leverage Web Application Firewalls (WAFs): Consider implementing a WAF to add an additional layer of protection and filtering for incoming HTTP traffic.
- Monitor and Respond to Security Incidents: Have an incident response plan in

place to detect and respond to security incidents effectively.

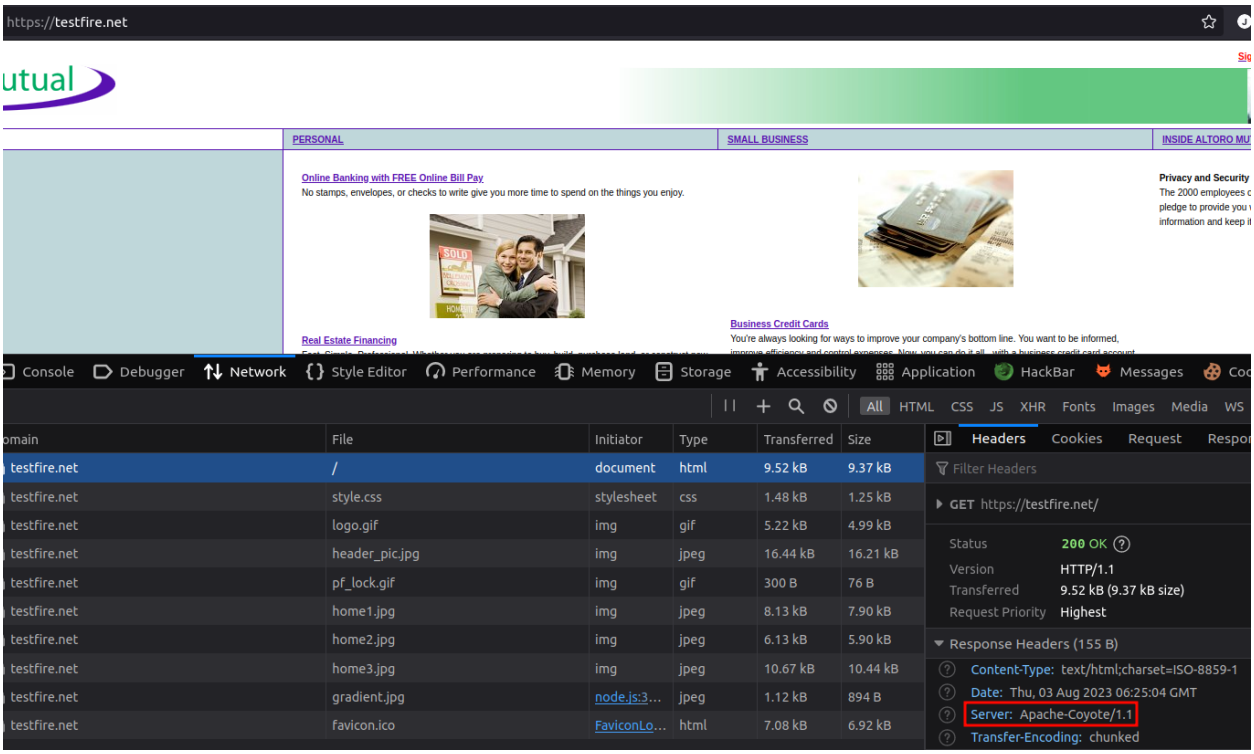
Remember, security is an ongoing process, and continuous monitoring and improvement are essential to maintain a secure web application. By implementing these recommendations, organizations can significantly reduce the risk of security breaches and enhance the overall security of their web applications.

## Finding-006: Leaking Server side information (Info)

| | |
|---|---|
| Description: | Security Misconfiguration refers to the improper implementation or configuration of security controls, which can leave an application or system vulnerable to attacks. This category includes a wide range of misconfigurations, such as default settings, unnecessary services or features enabled, and insufficient security settings. |
| CEW Code: | CWE-200: Exposure of Sensitive Information to an Unauthorized Actor |
| OWASP Category: | **A05:2021-Security Misconfiguration** moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category. |
| Affected URL: | https://testfire.net/ |
| References: | • https://cwe.mitre.org/data/definitions/200.html |

**Steps to reproduce:**

- Use browser's network tab to view HTTP response headers




## **Business Impact -** Very Low

- Aids attackers with additional information.

## **Recommendations**

- Disable Server Signature: Ensure that the server signature is not included in the HTTP response headers. This signature often contains information about the web server and its version, which could be exploited by attackers. In Apache, you can use the "ServerSignature" directive set to "Off" in the configuration file.
- Custom Error Pages: Configure custom error pages for different HTTP status codes. Avoid displaying detailed error messages to users, as they may expose sensitive information about your server and its configuration.

- Remove Sensitive Headers: Review and remove any unnecessary or sensitive information from the HTTP headers returned by your application or web server. Common sensitive headers include "Server," "X-Powered-By," and "X-AspNet-Version."
- Use Security Modules: Implement security modules or web application firewalls (WAFs) to filter and modify outgoing HTTP headers. These can help you remove sensitive information or sanitize headers before sending responses to clients.
- HTTP Strict Transport Security (HSTS): Enable HSTS to force users to connect via HTTPS. This helps protect against man-in-the-middle attacks and ensures all communication is encrypted.
- Content Security Policy (CSP): Implement a Content Security Policy to control what resources can be loaded and from where. This can help prevent malicious content injection and unauthorized data leaks.
- Application Security: Regularly audit your web application code for any potential vulnerabilities that may inadvertently expose sensitive information through HTTP headers. Follow secure coding practices and keep all libraries and frameworks up to date.
- HTTP Security Headers: Implement security-related HTTP headers like "X-XSS-Protection," "X-Content-Type-Options," and "Content-Security-Policy" to enhance your application's security posture.
- Regular Security Assessments: Conduct regular security assessments and penetration testing on your web application to identify and address any potential security flaws.

- Security Patching: Keep your web server, operating system, and other software components up to date with the latest security patches to minimize the risk of known vulnerabilities being exploited.

By following these recommendations, you can significantly reduce the chances of accidentally leaking sensitive server information through HTTP headers and enhance the overall security of your web application.

## Finding-007: Weak-cipher-suites:tls-1.0, 1.1 (Medium)

| | |
|---|---|
| Description: | Security Misconfiguration refers to the improper implementation or configuration of security controls, which can leave an application or system vulnerable to attacks. This category includes a wide range of misconfigurations, such as default settings, unnecessary services or features enabled, and insufficient security settings. |
| CEW Code: | CWE-327: Use of a Broken or Risky Cryptographic Algorithm |
| OWASP Category: | **A05:2021-Security Misconfiguration** moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it's not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category. |
| Affected URL: | https://testfire.net/ |
| References: | • https://cwe.mitre.org/data/definitions/310.html<br>• https://cwe.mitre.org/data/definitions/327.html |

**Steps to reproduce:**

- Nuclei Command to test: `nuclei -u http://testfire.net`

```
[tls-version] [ssl] [info] testfire.net:443 [tls10]
[tls-version] [ssl] [info] testfire.net:443 [tls11]
[tls-version] [ssl] [info] testfire.net:443 [tls12]
[weak-cipher-suites:tls-1.0] [ssl] [medium] testfire.net:443 [[tls10 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA]]
[weak-cipher-suites:tls-1.1] [ssl] [medium] testfire.net:443 [[tls11 TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA]]
[deprecated-tls] [ssl] [info] testfire.net:443 [tls10]
[deprecated-tls] [ssl] [info] testfire.net:443 [tls11]
```

## Business Impact - High

- Vulnerability to Attacks: TLS 1.0 is an outdated version of the TLS protocol and is known to have several vulnerabilities, including those related to its cipher suites. Weak cipher suites allow attackers to exploit vulnerabilities in the encryption and decryption process, potentially leading to data breaches,

unauthorized access, and data manipulation.

- Data Exposure: Weak cipher suites make it easier for attackers to intercept and decrypt sensitive data transmitted between clients and servers. This could include financial information, personal data, passwords, and any other confidential business data. In the event of a successful attack, the business could face legal and reputational consequences.
- Non-compliance: Using weak cipher suites in TLS 1.0 may result in non-compliance with industry standards and regulations, such as the Payment Card Industry Data Security Standard (PCI DSS) or General Data Protection Regulation (GDPR). Failure to comply with these regulations could lead to penalties and fines.
- Loss of Customer Trust: A data breach or security incident resulting from weak cipher suites can severely damage the business's reputation and erode customer trust. Customers may lose confidence in the company's ability to protect their data, leading to a loss of business and potential revenue.
- Limited Interoperability: As the industry moves towards using more secure versions of TLS, relying on TLS 1.0 with weak cipher suites can lead to interoperability issues with modern systems and applications. This may hinder the business's ability to collaborate with partners, vendors, and customers who have already upgraded to more secure protocols.
- Increased Risk of Downtime: Security incidents caused by weak cipher suites can disrupt business operations and result in downtime while the IT team responds to the breach and implements security measures. This downtime can be costly and negatively impact productivity.

## Recommendations

- Upgrade to a Secure TLS Version: Upgrade your TLS implementation to use TLS 1.2 or higher. TLS 1.0 is outdated and has known vulnerabilities, whereas TLS 1.2 and TLS 1.3 offer improved security and stronger cipher suites.
- Disable Weak Cipher Suites: Ensure that weak cipher suites are disabled on your servers and clients. By only allowing strong cipher suites, you can significantly reduce the risk of potential attacks.
- Regularly Update Software: Keep all software, including operating systems, web servers, and applications, up to date with the latest security patches. This helps to address newly discovered vulnerabilities and maintain a secure environment.
- Use Strong Encryption Algorithms: Implement strong encryption algorithms such as AES (Advanced Encryption Standard) for data encryption. Avoid older algorithms like RC4 and DES, which are considered weak and insecure.
- Enforce Perfect Forward Secrecy (PFS): Configure your servers to use Perfect Forward Secrecy (PFS). PFS ensures that even if an attacker compromises the server's private key, they cannot decrypt past communications.
- Monitor and Log Events: Implement comprehensive logging and monitoring of network traffic and security events. This will help you detect potential attacks and security breaches promptly.
- Perform Regular Security Audits: Conduct periodic security audits and penetration testing to identify vulnerabilities and weaknesses in your network infrastructure. Address any issues promptly to maintain a robust security posture.
- Educate Employees: Train your employees about best practices for data security, including the proper handling of sensitive information, recognizing phishing attempts, and following company security policies.
- Compliance with Industry Standards: Ensure that your security measures align with relevant industry standards and regulations, such as PCI DSS, GDPR, or any other applicable data protection laws.
- Implement Web Application Firewall (WAF): Consider deploying a WAF to

protect your web applications from various attacks, including those targeting weak cipher suites.

- Backup and Disaster Recovery: Regularly back up critical data and have a disaster recovery plan in place to mitigate the impact of potential data breaches or cyber incidents.
- Encrypt Data at Rest: Implement encryption for data at rest, such as databases and file storage, to provide an additional layer of protection for sensitive information.

By following these recommendations, your business can significantly enhance its security posture, protect sensitive data, and reduce the risk of cyberattacks and data breaches related to weak cipher suites in TLS 1.0. Remember that maintaining robust cybersecurity is an ongoing process, and it requires continuous vigilance and adaptation to address emerging threats.

# Thank You!

For any further clarifications/patch assistance, please contact:

*jagatarun9@gmail.com*